DOCUMENT RESUME

ED 152 287 IR 005 768

AUTHOR McGinnis, N. P.
TITLE CAL Authoring Aids.
INSTITUTION Alberta Univ., Edmonton. Div. of Educational Research Services.
REPORT NO RIR-76-3
PUB DATE Jun 76
NOTE 13p.; Paper presented at the Second Canadian Symposium on Instructional Technology (Quebec City, June 21-23, 1976)

EDRS PRICE MF-$0.83 HC-$1.67 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; *Computer Graphics; *Computer Oriented Programs; *Editing; Higher Education; *Instructional Design; Man Machine Systems; *On Line Systems; Programing

ABSTRACT

In their eight years of involvement in Computer Assisted Learning, the Division of Education Research Services has assisted many authors in developing their courseware. Whether the courseware has been a short half hour module or a large course involving over a hundred hours of student time, there have been many common problems. A major objective has been the reduction of the ratio of development and debug time to actual course execution time. This paper details some of the aids that have been developed and implemented to help authors produce better courseware in less time. One of the more important aids discussed is an on-line editing capability, which can be used to correct format and spelling errors in displays while students are actually on-line taking the course. Another major aid is the drawing system. This sub-system timeshares with other authors developing courseware, and students taking courses. It allows authors to create drawings and graphics for themselves and other authors, to merge several sets of drawings, and to produce hardcopy documentation of the drawings. Other aids already incorporated into the system and some foreseen for future implementation are also mentioned. (Author/VT)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED152287

U S DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

RIR-76-3

CAL AUTHORING AIDS

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

E.W. Romaniuk

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC) AND USERS OF THE ERIC SYSTEM."

Presented to: Second Canadian Symposium
on Instructional Technology
Quebec City, P.Q.
June 21-23, 1976

N.P. McGinnis

Division of Educational Research Services
Faculty of Education, The University of Alberta
Edmonton, Alberta T6G 2G5

IR003768

CAL AUTHORING AIDS

ABSTRACT

The Division of Educational Research Services (DERS) at the University of Alberta has been actively involved in Computer Assisted Learning for 8 years. During this period DERS has had the opportunity to assist many authors in developing their courseware. Whether the courseware has been a short half hour module, or a large course involving over a 100 hours of student learning time there have been many common problems. A major objective has been the reduction of the ratio of development and debug time to actual course execution time (i.e., 50 to 100 hours of preparation time required to produce one hour of instructional sequence). This paper will detail some of the aids that have been developed and implemented to help authors to produce better courseware in less time.

One of the more important aids to be discussed will be an on-line editing capability. An important feature of this editing capability is that it can be used to correct format and spelling errors in displays while students are actually on-line taking the course.

Another major aid to be discussed will be the drawing system. This sub-system timeshares with other authors developing courseware, and students taking courses. It allows authors to create drawings and graphics for themselves and other authors, and to merge severals sets of drawings, and to produce hardcopy documentation of the drawings.

Other aids already incorporated into our system and some we foresee for future implementation will also be mentioned.

INTRODUCTION

The Division of Educational Research Services (DERS) at the University of Alberta has been actively involved in Computer Assisted Learning (CAL) for over 8 years now. During this period of time we have witnessed CAL pass through various phases. DERS continues to operate with an IBM 1500 computer which was one of the earlier attempts at the complete CAL system. Other users later adopted subsystems attached to larger machines such as Coursewriter III and the CAN system. Today we are all witnessing the emergence of the Plato system, which has a large centrally located cpu with the capability to drive many remote terminals. Also creating much interest is the TICCIT system which is designed for a smaller cpu driving a small number of terminals with little remote capability. Here in Canada, we have the development of the NATAL language through the guidance of the National Research Council and the embryonic growth of a CAL industry. All of this cannot help but to encourage further work and research.

DERS has certainly attempted to keep on top of the latest developments. But at the same time we have been attempting to set some trends which we feel to be important. Initially the DERS system was used in the main for the research purposes of the educational psychologists. Since they were already beginning to use the computer for the processing of their data, the DERS system was the perfect complement to aid them in the collection of their data. They merely had to program into the machine their various tests and the terminal became their data collecting tool. With the availability of light pens, keyboards, film projectors, audio units, and character and graphic display under computer control, they had a very rich medium in which to present their tests for research. Meanwhile, other users more interested in CAL were busy getting their feet wet by programming modules of instruction. Most of these modules were of fairly short duration, 10 to 30 minutes. Much time was spent trying to solve a multitude of problems such as optimizing display techniques, developing answer analysis algorithms, setting up operating and documentation standards, and many more.

We have yet to solve all our problems but today DERS has one distinct trend. Gone are most of the educational psychologists using the system for data acquisition. Also we have very few users making use of the short modules of instruction that are available. Instead we are concentrating on the longer courses. The students taking these courses can spend anywhere from 20 to 100 hours and more to complete a particular course. Examples of some of these courses are:

1) An introductory level French course requiring approximately 25 hours to complete.

2) A graduate level statistics course requiring approximately 80 hours for the average student.

3) An early chilhood development course for teachers which was received from Pennsylvania State University taking about 50 hours to complete.

4) A cardiology course for 2nd year medical students taking about 45 to 50 hours.

5) A basic electronics course, received from the U.S. Army, taking about 35 hours to complete.

The common denominator for these courses with the increased length, became the need for a more systemized group approach to the preparation of a course. We were able to observe the needs of authors from this group approach, and subsequent requests evolved some authoring aids which will now be elaborated upon.

THE EDIT COMMAND

Until last year, to correct errors in formatting or spelling, an author had to replace a complete line of code. Since the Coursewriter II system is interpretive, the software readily allows for this, but authors find it to be an inefficient method. Up to 240 characters might have to be re-typed to correct the simplest of spelling errors.

Now thanks to the programming expertise of Norman Margolus, a student at our university, we have corrected this deficiency with the edit station command. It is similar to other file editing capabilities that we have all utilized on other systems. However, there are some special features to our edit command which we feel to be noteworthy.

First of all, it is possible to interrupt the listing or execution of a course, edit in some changes, and then resume the listing or execution of the course. In particular, while listing part of a course at a crt, an author can interrupt to edit in some changes, and the system will then resume the listing at the statement edited to maintain continuity. While executing a course an author can interrupt the execution to correct spelling errors, to replace a phrase or to alter the location of the display on the crt. After editing, the author can simply enter the command GO and the system will resume execution of the course one or more frames prio to the edited change. This allows the author to re-execute through the edited material to ensure the proper corrections have been made. This editing procedure has even been used by authors during their initial testing sessions with students on-line. It is at this time that the myriad of possible paths for execution of a program begin as students enter their varied responses. It is also at this time that some of the errors along these paths will show up. Along with the LABEL command the author at his terminal can quickly correct many of the errors.

Without getting too technical, some of the finer details of the edit command are:

1) An author can refer to the label sequence of the line of code to be edited. Alternatively, if he is listing or inserting new lines of code, he can use an '*' (asterisk) which will specify that the last line listed or inserted is the one to be edited.

2a) The author can then specify that the command should SCAN through n lines of code to attempt to locate the material to be edited. This feature is useful when executing code and an error is noted within a particular frame. The author merely has to use this feature, rather than having to know the exact label sequence of code where the error exists.

2b) Another choice for the author instead of SCAN is REPEAT. In this case the command will 'repeat' through the number of lines of code specified, making the requested change whenever the code to be edited is located. This has been useful in making wholesale changes in courses received from other locations. For example, with the basic electronics course, it was necessary to change all the references to an American Army meter for a meter that was readily available in the University's electronics' lab. To have someone visually scan through several thousand pages of code would have been a monumental task. The editor was able to do this work in short order.

2c) In most situations, the author will default the use of the SCAN or REPEAT options. The edit command will then only execute upon the label sequence (or '*') specified.

3) if the author 'enters' the command in a normal manner and if the code to be edited is located, the operation code, the label sequence number of the statement and the information to be replaced will be displayed on the bottom line of the crt. The cursor will be positioned on the line above and the author is then expected to enter the replacer. The author has the option of cancelling the editing at this point without any changes being effected. The author would be free to resume whatever activity he interrupted (ie. listing, inserting, etc.). In the normal case when the author enters a replacer, the newly edited material will be displayed with the changes on the second last line of the crt. The cursor will be presented on the bottom line of the crt. The author has the option to type in a new edit command, another station command, or to simply 'enter' which will return him to the authoring activity he interrupted.

4) If the author 'enters' his edit command in a special manner (alternate code enter) the edit function will not request a replacer from the author. Instead it will use the last replacer entered by the author. This is normally used in conjunction with the repeat feature.

5) The edit command 'remembers' parameters. Therefore the author can omit various parameters when they continue to be the same as he has already specified. This is very time saving, especially when one is not a good typist.

At the moment, some of the shorter Coursewriter II statements cannot be edited. They must be 'replaced' (an authoring command combining 'delete' and 'insert' logic). The reason these statements cannot be edited is that the system converts them from text code to binary code for the interpreter. We are considering changing this process so that all code within courses would be in text code, with the binary code eliminated. All statements could then be edited. Interestingly enough these changes would then also allow us to edit all op codes and parameters. We are speculating that this would allow us to easily edit our courses into a format that would resemble the NATAL language specifications. Our courses, of course, could then become much more transportable than they now are.

THE LABEL COMMAND AND THE GO COMMAND

Reference was made while describing the edit command to 2 other commands; the LABEL command and the GO command.

The LABEL command serves to tell a user where he is in the course. We have other tools for determining a student's location but this one gives us the most flexibility and is easiest to use. The follwing points detail the information that this command provides:

1) Segment number (analagous to chapter number).

2) The label sequence number of the current instruction being executed.

3) The label sequence number of the last executed pr. A pr location is usually the beginning of a concept or a problem.

4) The label sequence number of the last executed question.

The GO command allows the user to override the normal implicit and explicit branching in Coursewriter and to specify branching to the following locations:

1) A valid label sequence number.

2) The last executed pr location.

3) The last executed question.

The LABEL command is initiated from the bottom line of the crt and the results are presented on this same line. If a student gets into trouble, a proctor (class supervisor) can make use of these two commands at the student's station. If the student is having trouble understanding the content the proctor can attempt to explain it. The proctor may choose to use the GO command to review a few previous frames with him. If it becomes obvious that there is a logical bug in the program, or that the content is incorrect, then the LABEL command information can be an adjunct in documenting the error. The error can then be presented to the author for corrective action.

In the future we will not be able to expect all proctors to be proficient in the content of all courseware on a CAL system. We are planning to have the correct answers to all questions as documentation in label sequence number order. Referring to this type of information and using the LABEL command, the proctor should be able to assist any student who comes to an impasse because of his inability to absorb the concept or because of simple programming deficiencies. If the student was having learning problems he can approach his instructor at a later time. If it was a programming error the proctor can notify the author of the situation.

THE DRAWING SYSTEM

The IBM 1500 system has always had the capacity to present graphic material on the crt. This could be anything from a set of Chinese characters to a cartoon. However, there were some drawbacks that came to light when users became proficient in developing these character sets. First of all they were tedious to produce as it required transferring the intended image to a punched card. Some of the cards ended up with the appearance of lace doilies. More importantly, courses that utilized graphics heavily became mutually exclusive to each other. This means that some courses could not be run simultaneously on the system while other users were executing the other courses. This created queuing problems and scheduling problems for users.

Several years ago we modified the system to remove these constraints. A document detailing how this was done is entitled "Macro-Graphics for the 1500 System". Basically the system allowed users to develop graphic material right at the terminal. The program to do this allowed for the following types of options:

1) Erase all or selected portions of a graphic.

2) Copy from one part of the screen to another (the option also exists to copy material from another station).

3) Rotation of the material 180 degrees on the horizontal or vertical axis. This is called 'invert' and 'mirror' respectively.

4) Specify the exact boundaries of a new graphic.

Now several new features are available:

5) User definition of three points on the crt - P1, P2 and P3.

6) Given that points P1 and P2 are defined, be able to join the points with a line, or to create a circle with the two points being the extremeties of the diameter of the circle.

7) Given that all three points are defined, to produce an arc beginning at P1, running through P2 and ending at P3.

8) The user can respecify the location of the points at any time to produce new lines, circles and arcs.

Normally when a line or arc crosses an area already lit on the screen, it does not increase the intensity of light and become more visible. By using the 'exclusive-or' option this problem can be overcome. Dots along the line or circle that are not already lit will be lit. Dots that are already lit will be turned off. This produces a dark line through an already lighted area. It is an excellent way to maintain the continuity of the intended path in the graphic.

Once a graphic is produced, it is stored on-line as a macro. This means that it is available to all authors to copy into their respective courses. In the meantime, we send a special tape to a larger computer on campus, where the new graphic is plotted on paper for documentation purposes. Information detailing the graphic is stored in a 'kwic' index so that others can easily reference and view finished graphics.

The drawing system has enhanced the authors ability to convey information that would be difficult or lengthy to express in text. Also incorporating humor into some of the graphics has enabled an author to provide some personality into his course at strategic moments.

OTHER AUTHORING FEATURES

Some prior reference in this paper has been made to macros. A macro can be loosely defined as a framework of code into which parameter substitution can be made to tailor the instructions to an author's needs. However, what frequently occurs at DERS is that the author decides after the fact that some of his already developed instructional material should have been designed for use as a macro. From this we have developed the ability to copy course code from other parts of one's own course, or from other courses. The author while inserting material at a terminal can copy material from elsewhere and have it fit right into the mainstream of his coding. He looses little time in doing this, and can then use the edit command to make the changes he deems necessary for his course.

We still have some problems to contend with in copying. Presently all courses are available for on-line copying by all authors. Because of our open shop atmosphere we do not have controls on what can and cannot be copied, except that it must be an author doing the copying. We will soon have to contend with courses of a restricted nature such as exams. A hierarchy of controls for course availability will have to be developed.

The statistics course and the basic electronics course developed our need for a CALCULATOR command. The student now has at the terminal a simple calculator with similar power to that of the cheaper models that are currently available. It allows for expressions including addition, subtraction, multiplication, division, exponentiation and parentheses. Again the bottom line of the crt is utilized for accepting the expression to be calculated. The result will be displayed right-justified on the same line. Should the calculator find an error in syntax in the expression, an appropriate error message will be displayed. There are approximately 20 possible error situations allowed for, from 'dividing by zero' to 'mis-matched parentheses'.

We foresee several more features for the calculator. Trignometric capability, matrix manipulation, and 'named-variable specification' are some examples. On the other hand we can foresee authors wanting to control the availability of the calculator during exams and certain learning situations. This has been provided for, but not yet implemented.

One item which the author does not normally have to contend with, but is there to assist him is an accounting system. Records are maintained that give us information about each user's session at a terminal, whether it be several times a day or once a month. The information includes; user id, course name, terminal number, date, time, amount of time signed on, and a project number. All of this enables us to present authors and proctors with information about system usage and utility. The proctor can quickly determine who is progressing quickly enough to finish the course within the delegated time. He can also use it to determine attendance if this is a required item.

More importantly to us, the information helps us to chart out the scheduling of the system for future semesters. Some users do request more time that is really needed and availability of the information helps us to convince users of this point. Also because we were able to chart this information, and from this determine a need, the system is now available on a free-time basis to users on the weekend. The system is now available from 8 to 4 on Saturdays and from 1 to 5 on Sundays.

Hardware reliability has always been a concern. Mechanical components such as the audio unit and the film projectors have performed well, but they do give the occasional problem. The system was designed to output a numeric message at the terminal when a component malfunctions occurs. Most users found these messages incomprehensible and were forced to seek a proctor for assistance. We have improved the human interface problem a little by having the computer output a more meaningful description for some of the problems. Now the system may tell the user to 'focus his film projector' or that the user must 'load an audio tape'. Students have been able to react to these messages, freeing up the proctor to handle some more difficult problems.

In this vein we are contemplating another change. At present an audio unit or a film projector can fail to perform as instructed. This can cause a student to miss an important fact, making it difficult for him to answer questions and proceed in the course. We would like to give the student the ability to repeat unit functions, so that they could view a slide again, or repeat a message that did not come through clearly the first time. This should not be too difficult to accomplish, but again certain situations, such as exams where repetition may be frowned upon, will have to be accounted for.

CONCLUSIONS

This paper demonstrates that we have not been adverse to jumping into the operating system to make changes. In all cases while attempting to improve the system, we have maintained the systems compatibility that is so necessary for existing production courses. We feel that our progress is helping to make the computer a more effective tool for teaching and for education.

REFERENCES

DERS COMPUTER ASSISTED INSTRUCTION BULLETIN, Division of Educational Research Services, University of Alberta.

IBM. IBM 1500 Coursewriter II, Author's Guide, (GY26-1580-0), San Jose, California, Author, 1968.

McGinnis, N.P. Macro-Graphics for the 1500 System, Presented at the ADIS Conference, Quebec City, Quebec, August, 1972.